Parallel computer network for multiple instructions and data - using common transmission rail to provide authorised access by all central processing units to data distributed throughout network

Patent number: DE4221278
Publication date: 1994-01-05
Inventor: VORBACH MARTIN (DE)
Applicant: VORBACH MARTIN (DE)
Classification:
- international: G06F15/16
- european: G06F15/17
Application number: DE19924221278 19920629
Priority number(s): DE19924221278 19920629

Abstract of **DE4221278**

The parallel computer network includes several units (2) consisting of a CPU (3), a random-access memory (4) and a bus control unit (5) linked by an internal data bus (6). The units are interconnected by a data transmission rail (7).
The bus control unit and CPU feature an additional connection for exchange of status information so that the CPU can have access to data from any unit (2) of the network. The RAMs are protected against unauthorised access by association of individual storage cells with additional control stores.
ADVANTAGE - Data exchange between individual components of the network is performed without the programmer's intervention and with automatic data synchronisation.

Data supplied from the ***esp@cenet*** database - Worldwide

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 42 21 278 A 1**

(51) Int. Cl.[5]: G 06 F 15/16

(21) Aktenzeichen: P 42 21 278.2
(22) Anmeldetag: 29. 6. 92
(43) Offenlegungstag: 5. 1. 94

DE 42 21 278 A 1

(71) Anmelder:
Vorbach, Martin, 7500 Karlsruhe, DE

(74) Vertreter:
Zahn, R., Dipl.-Ing., Pat.-Anw., 76229 Karlsruhe

(72) Erfinder:
gleich Anmelder

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Parallelrechnernetzwerk

DE 42 21 278 A 1

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

BUNDESDRUCKEREI 11. 93 308 061/358

Beschreibung

Die vorliegende Erfindung bezieht sich auf ein Parallelrechnernetzwerk nach dem Oberbegriff des Patentanspruchs 1.

Ein gattungsgemäßes Parallelrechner- oder Datenverarbeitungsnetzwerk ist Gegenstand des Patents ... (Patentanmeldung P 41 27 192.0). Dabei sind jeweils mehrere CPU in Gestalt von Systemeinschüben gruppenweise zusammengefaßt und jede dieser CPU-Gruppen ist über einen PCU mit einem Datenbus gekoppelt und so im Gesamtsystem integriert. Die dabei zugrunde gelegten CPU sind jeweils so zu verstehen und konfiguriert, daß der notwendigerweise zugehörige Speicher, also der RAM, über einen internen Datenbus mit der CPU gekoppelt ist, und zwar jeweils so, daß von dem PCU aus betrachtet die RAM den eigentlichen Prozessoren, also den CPU, nachgeordnet sind. Dies hat letztlich zur Folge, daß bei einem Zugriff einer CPU auf den Speicher einer beliebigen anderen CPU, um etwa Daten oder Variable zu adressieren, dieser Zugriff über die LINKS oder Schnittstellen der CPU erfolgt beziehungsweise abgewickelt werden muß. Die Folge davon ist, daß diese funktionalen Verbindungen beziehungsweise Speicherzugriffszyklen nicht nach dem sogenannten Shared-Memory-Prinzip, bei dem sich mehrere CPU gleichberechtigt einen gemeinsamen Speicher teilen, möglich sind, da die CPU aktiv in den Ablauf eingreifen müssen. Vielmehr ist es hierbei Aufgabe des Programmierers für den richtigen Ablauf, d. h. die richtige Arbeitsweise des Mehrrechnersystems, zu sorgen; gleichermaßen muß auch die erforderliche Datensynchronisation vom Programmierer bewältigt werden.

Die der vorliegenden Zusatzerfindung beziehungsweise Weiterbildung des dem Patent ... (Patentanmeldung P 41 27 192.0) entsprechenden Datenverarbeitungsnetzwerks zugrunde liegende Aufgabe besteht darin, ein Parallelrechnernetzwerk der gattungsgemäßen Art anzugeben, bei dem der Datenaustausch zwischen den einzelnen Komponenten ohne Vorgabe durch den Programmierer der Anlage erfolgt und bei dem die Datensynchronisation weitestgehend autark und automatisch erfolgt.

Diese Aufgabe wird durch die im kennzeichnenden Teil des Patentanspruchs 1 angegebene Verschaltung beziehungsweise Kopplung der dem Parallelrechnernetzwerk zugrunde liegenden Funktionselemente CPU, PCU, RAM und Datenbus gelöst.

Eine besondere Ausgestaltung im Hinblick auf eine sichere Arbeitsweise der RAM sowie zur Vermeidung von Synchronisationsproblemen ist Gegenstand des Patentanspruchs 2.

Abgesehen von dem bereits aufgabengemäß bedingten Vorteil, daß sich der Programmierer nicht um die Verteilung der Daten beziehungsweise Variablen innerhalb des Parallelrechnernetzwerks zu kümmern braucht, besteht ein ganz besonderer Vorzug des erfindungsgemäßen Parallelrechnernetzwerks in der erhöhten Virensicherheit. Da nämlich ein Virus grundsätzlich eine andere Programmspezifikation aufweist, als die der anderen Programme, kann er auch nicht auf die Daten beziehungsweise Variablen und den Code dieser anderen Programme zugreifen und Unheil anrichten. Ganz allgemein zeichnet sich das im Vorstehenden charakterisierte und im Detail nachfolgend zu beschreibende Parallelrechnernetzwerk dadurch aus, daß innerhalb dieses Parallelrechnernetzwerks die Möglichkeit besteht, Daten im gesamten Parallelrechnernetzwerk sinnvoll zu verteilen ohne daß für den Programmierer eine wesentliche Belastung bezüglich der Kommunikation entsteht — schließlich kann von allen CPU gleichermaßen und gleichberechtigt auf die verteilten Daten zugegriffen werden.

Das erfindungsgemäße Parallelrechnernetzwerk wird im folgenden anhand der Zeichnung näher erläutert. Diese zeigt in

Fig. 1 eine Schemadarstellung eines aus einer Mehrzahl von sogenannten Parallelrechnereinheiten aufgebauten Parallelrechnernetzwerks;

Fig. 2 eine Schemadarstellung zur Erläuterung der Adressierung der Mehrzahl der Parallelrechnereinheiten;

Fig. 3 eine Schemadarstellung eines RAM zur Erläuterung der Zugriffssicherheit.

Fig. 1 zeigt ein Parallelrechnernetzwerk **1**, das aus einer Vielzahl parallel zueinander angeordneter sogenannter Parallelrechnereinheiten **2** besteht beziehungsweise konfiguriert ist. Unter einer Parallelrechnereinheit ist im gegebenen Zusammenhang eine Recheneinheit eines Rechners zu verstehen, der auf der Grundlage einer Mehrdaten- und Vielfachinstruktionsmaschine (MIMD) basiert. Diese Mehrdaten- und Vielfachinstruktionsmaschine beziehungsweise Recheneinheit besteht jeweils aus einem Prozessor **3**, der sogenannten CPU, einem Speicher **4**, dem sogenannten RAM, und einer Bussteuereinheit **5**, der sogenannten PCU, welche die Verbindung eines internen Datenbus **6** und einer Datenübertragungsschiene (DÜ-Schiene) **7** steuert.

Die Recheneinheit ist in der Lage Programme, gegebenenfalls auch nur Programmabschnitte abzuarbeiten und kann — und zwar entsprechend der erfindungsgemäßen Konfiguration — über den internen Datenbus **6** sowohl auf den jeweils eigenen RAM **4**, als auch — über die PCU gesteuert — auf den RAM einer anderen parallel angeordneten Parallelrechnereinheit **2** zugreifen.

Die auf einer Parallelrechnereinheit eigenständig laufenden Programme beziehungsweise Programmabschnitte werden im weiteren als TASK bezeichnet, deren mehrere auf einer Parallelrechnereinheit **2** quasi gleichzeitig abgearbeitet werden können. Jedem TASK ist eine eindeutige Identifikationsnummer zugewiesen, beziehungsweise zugeordnet, die selbstverständlich nur einmal innerhalb des Parallelrechnernetzwerks **1** existieren darf. (Dies ist von besonderer Wichtigkeit, da in der vorliegenden Beschreibung die Identifikationsnummer der TASK zum Schutz der taskeigenen Daten, also der Daten die keinesfalls von einem anderen TASK benutzt werden dürfen, verwendet wird).

In Verbindung mit dem vorliegenden Parallelrechnernetzwerk kommt der Frage der eindeutigen Zugriffssicherheit eine besondere Bedeutung zu. Dieses im nachfolgenden noch zu beschreibende LOCKING gewährleistet, daß ein RAM **4** gegen einen illegalen oder unzulässigen Zugriff geschützt ist.

Für die Funktionsweise des Parallelrechnernetzwerks ist ferner eine als Taskswitcher oder Scheduler bezeichnete Einheit von Bedeutung, die — für gewöhnlich im Betriebssystem integriert — die Umschaltung der aktiven TASK gewährleistet. Dadurch wird zwar der Eindruck erweckt, daß mehrere TASK gleichzeitig verarbeitet werden — dieser Eindruck ist jedoch falsch, da die TASK nacheinander abgearbeitet werden. Die genannte Umschaltung geschieht eben durch diesen Taskswitcher.

Da der Taskswitcher eine für die Funktionssicherheit des Parallelrechnernetzwerks wichtige Rolle spielt und

er weiterhin auch sehr schnell sein sollte und auf einen sogenannten, noch zu erläuternden DENIED-Zugriff reagieren muß, sollte er (muß aber nicht) hardwaremäßig implementiert sein. (Somit sollte aus Sicherheitsgründen die TASK-Identifikationsnummer hardwaremäßig vorgegeben werden).

Die Konfiguration und Funktionsweise des Parallelrechnernetzwerks 1 wird — auf der Grundlage der Fig. 1 und der vorstehenden Begriffsspezifikationen — im Nachfolgenden näher erläutert:

Die Verschaltung beziehungsweise Verdrahtung der Mehrzahl von Parallelrechnereinheiten 2 geht von den genannten PCU 5 aus, die direkt, d. h. unmittelbar über eine gewöhnliche DMA-ähnliche beziehungsweise -analoge Zugriffsfunktion (DMA = Direct Memory Access) im RAM 4 der jeweiligen über einen gemeinsamen Datenbus 7 gekoppelten Parallelrechnereinheiten 2 arbeiten, also die Busmasterfunktion für den internen Datenbus 6 übernehmen können. Die genannte Busmasterfunktion wird durch Signale BUS-REQUEST (BREQ) und BUS-ACKNOWLEDGE (BACK) zwischen den Einheiten CPU 3 und PCU 5 gesteuert, wobei das Signal BREQ die Busfreigabeanforderung an die CPU 3 bedeutet und das Signal BACK die Busfreigabe signalisiert.

Anders als beim Datenverarbeitungsnetzwerk des älteren Patents ... (Patentanmeldung P 41 27 192.0) können die einzelnen Parallelrechnereinheiten 2 des Parallelrechnernetzwerks 1 über ihre konjugierten PCU 5 also jeweils direkt auf die RAM 4 der parallelen Parallelrechnereinheiten 2 zugreifen — die funktionalen Verbindungen brauchen also nicht mehr unter Zuhilfenahme der CPU 3 abgewickelt zu werden.

Das Ganze setzt insoweit jedoch auch eine modifizierte Adressierung voraus, und zwar wird jeweils zusätzlich zur Adresse einer Parallelrechnereinheit 2 (HEADER eine Adresse) noch die gewünschte Speicheradresse (Segment) übertragen, aus der Daten beziehungsweise Variable gelesen beziehungsweise in die Daten beziehungsweise Variable eingeschrieben werden sollen. Jeweils nach Ablauf eines Zugriffs, d. h. eines Speicherzyklus, erhält die den RAM 4 beanspruchende CPU 3 ein Bestätigungssignal, ein Fehlersignal oder ein DENIED-Signal.

Der Prozessorspeicher, d. h. der RAM 4, ist als sogenannte lineare Speicherbank organisiert. Gemäß Fig. 2 besteht eine Adresse 10 dabei aus einem Segment 11 (A_0 ... A_i) und dem sogenannten HEADER 12 (A_{i+1} ... A_n). Der HEADER 12 enthält die Adresse der gewünschten Parallelrechnereinheit 2; das Segment 11 enthält die gewünschte Adresse innerhalb des adressierten Speicherraums. Ist der HEADER 12 dabei gleich der Adresse der aktuell zugreifenden CPU 3, so findet der Zugriff innerhalb des eigenen RAM 4 statt (vergleiche Pfeil X in Fig. 1); ist die Adresse ungleich, so wird die PCU 5 angesteuert und dadurch der HEADER 12 an die DÜ-Schiene 7 übergeben, gleichermaßen auch das Segment 11 ... sie dienen als Adresse für den adressierten RAM (vergleiche Pfeil Y in Fig. 1) und die Parallelrechnereinheit.

Der weitere Zugriff ist wie folgt: Die durch einen HEADER 12 angesprochene Parallelrechnereinheit 2 (dies ist anders als beim Patent ... (Patentanmeldung P 41 27 192.0) nicht eine CPU 3) gibt den Speicherzugriff auf die übertragene Adresse frei, sofern aufgrund eines LOCKING kein Widerspruch entsteht. Nun kann die entsprechende CPU frei über den Zielspeicherbereich in der Parallelrechnereinheit verfügen, insbesondere den Zielspeicherbereich für sich LOCKEN, sofern ein sogenannter Read-Modify-Write-Zyklus vorliegt. Ist der Zyklus vollständig beendet, so gibt die entsprechende CPU den Speicherblock der adressierten Parallelrechnereinheit 2 wieder frei. Zu bemerken ist insofern noch, daß die adressierte Parallelrechnereinheit auch während eines Zugriffs auf ihren RAM 4 frei über denselben verfügen kann, sofern einerseits kein Buskonflikt vorliegt, und andererseits sie nicht Speicherbereiche anspricht, die etwa von anderen Parallelrechnereinheiten geLOCKt sind.

Stößt eine Parallelrechnereinheit 2 auf ein Speicherwort, das von einem anderen TASK als dem der zugriffswilligen Parallelrechnereinheit geLOCKt ist, so kann diese Parallelrechnereinheit nicht auf das Speicherwort zugreifen. Die entsprechende Parallelrechnereinheit erhält über die DÜ-Schiene 7 ein DENIED-Signal zurück. Der Taskswitcher dieser Parallelrechnereinheit 2 schaltet nun sofort auf einen anderen lauffähigen TASK um, so daß letztlich die CPU keine Zeit durch Warten auf eine Speicherfreigabe verliert.

Zu bemerken ist, daß die Funktion des Taskswitcher eines der wichtigsten Merkmale des erfindungsgemäßen Parallelrechnernetzwerks ist. Grundsätzlich wird jeder Aufruf auf eine Speicherstelle, die a) nicht jedem TASK frei zur Verfügung steht (vergleiche PUBLIC gemäß Fig. 3) beziehungsweise b) nicht dieselbe TASK-Identifikation wie die aufrufende TASK hat, mit der sofortigen Unterbrechung des aufrufenden TASKs quittiert. Der TASK wird jedoch bei einer erneuten Aktivierung nochmals mit demselben Aufruf auf dieselbe Speicherstelle gestartet, d. h. der TASK läuft erst an, wenn die aufzurufende Speicherstelle freigegeben (PUBLIC) ist. Durch diese Technik wird nicht nur die Verwaltung der Daten beziehungsweise Variablen organisiert, sondern es werden analog auch die TASK-Aufrufe gesteuert; TASK werden durch das Setzen eines Startbefehls in einer freien (PUBLIC) Speicherstelle aufgerufen. Der TASK testet die Speicherstelle — enthält sie den Startbefehl, so startet er, ansonsten deaktiviert er sich selbst und zwar durch einen Befehl an den Taskswitcher. Dem aufgerufenen TASK wird eine Antwortspeicherstelle des aufrufenden TASKs — die frei zugänglich (PUBLIC) ist — übergeben. Dort setzt er ein Endesignal, sobald der TASK erfolgreich terminiert (beendet) ist. Der aufrufende TASK testet auf dieses Endesignal, sofern eine TASK-Synchronisierung erforderlich ist.

Ist das Endesignal nicht gesetzt, so deaktiviert sich der TASK durch einen Befehl an den Taskswitcher selbst und startet erneut mit dem Test des Endesignals.

Um letztendlich die Aktualität der Daten beziehungsweise Variablen zu gewährleisten, insbesondere jedoch um zu verhindern, daß während eines sogenannten Read-Modify-Write-Zyklus falsche Informationen gelesen werden (während eines Modify könnte eine andere CPU die Daten lesen, welche allerdings bereits durch den Modify auf einer anderen CPU nicht mehr aktuell sind) ist ein Speicher-LOCK-Mechanismus vorgesehen. Hierzu werden — vergleiche Fig. 3 — neben den üblichen RAM 4 sogenannte Kontrollspeicher 14 implementiert, die Auskunft über die Zugriffsrechte geben. In diesem Kontrollspeicher 14 werden die Identifikationsnummern der TASK abgelegt, die den jeweiligen Speicherelementen zugeordnet sind; darüberhinaus enthält dieser Kontrollspeicher 14 Informationen darüber ob das Speicherelement für Fremdzugriffe freigegeben ist oder nicht. (Variablen die für Fremdzugriffe freigegeben sind, werden innerhalb der dazugehörenden TASK als PUBLIC bezeichnet — dieses Verfahren ist durch objektorientierte Programmiersprachen — vergleiche OOPS

– bekannt).

Beispielsweise kann über eine – vergleiche a) in Fig. 3 – aus lauter "0" bestehende "Date" angegeben sein, daß der als konjugierte Speicherplatz einem Zugriff offen, also die Variabel PUBLIC deklariert ist. Gemäß der in b) dargestellten Daten ist angegeben, daß der Speicher durch das Betriebssystem geLOCKt ist.

Das Beispiel c) zeigt, daß ein TASK mit der Identifikationsnummer "2" gerade auf den konjugierten zugreift; im Beispiel d) sei gezeigt, daß prinzipiell die Möglichkeit besteht, Variablen für Gruppen (Programme) zu LOKKEN, indem zum Beispiel das höchstwertige Bit angibt, ob eine Gruppe (Gruppenidentifikation) oder eine TASK-Identifikation vorliegt.

Grundsätzlich stehen im System auch erweiterte Speicherzugriffe zur Verfügung:

READ LOCK: Lesen und LOCKEN des Speichers (Einschreiben der eigenen Task-IDENTIFIKATION)
WRITE LOCK: Schreiben und LOCKEN des Speichers (Einschreiben der eigenen Task-IDENTIFIKATION)
READ UNLOCK: Lesen und UnLOCKEN des Speichers (Löschen der eigenen Task-IDENTIFIKATION)
WRITE UNLOCK: Schreiben und UnLOCKEN des Speichers (Löschen der eigenen Task-IDENTIFIKATION)

Bei jedem Zugriff wird dabei getestet, ob der Speicher nicht von einem anderen Task gesperrt ist.

Patentansprüche

1. Aus einer Vielzahl von (im folgenden CPU genannten) Prozessoren, (im folgenden RAM genannten) Speichern und (im folgenden PCU genannten) Bussteuereinheiten zur funktionalen Verbindung der CPU's über eine (im folgenden DÜ-Schiene genannte) gemeinsame Datenübertragungsschiene bestehendes Parallelrechnernetzwerk nach Patent (Patentanmeldung P 41 27 192.0), **dadurch gekennzeichnet,**
daß je eine CPU (3), ein RAM (4) und ein PCU (5) eine Mehrdaten- und Vielfachinstruktionsmaschine (MIMD) bilden, und
daß eine Vielzahl der Mehrdaten- und Vielfachinstruktionsmaschine (Parallelrechnereinheiten 2) über die DÜ-Schiene (7) funktional derart miteinander gekoppelt sind, daß die DÜ-Schiene (7) unmittelbar mit den PCU's (5) und diese über einen internen Datenbus (6) einerseits mit der konjugierten CPU (3) und andererseits mit dem konjugierten RAM (4) verbunden sind, wobei PCU (5) und CPU (3) eine zusätzliche Verbindung zum Austausch von Statusinformationen aufweisen, so daß die CPU (3) auf die Daten beziehungsweise Variablen jeder beliebigen Parallelrechnereinheit (2) zugreifen kann.

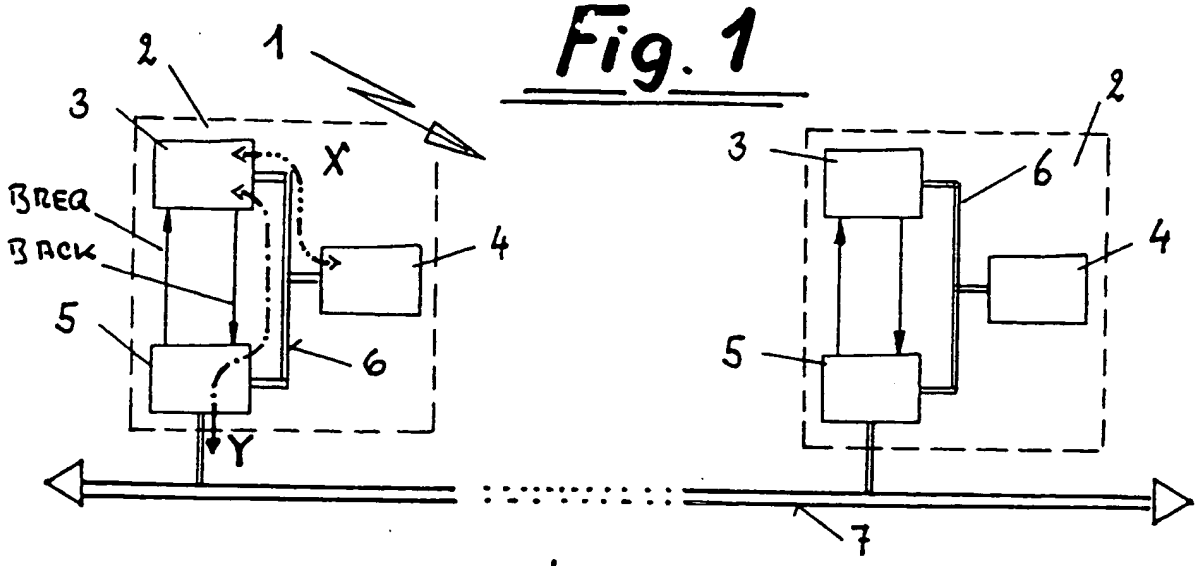
2. Parallelrechnernetzwerk nach Anspruch 1, dadurch gekennzeichnet, daß die RAM's (4) gegen einen unberechtigten Zugriff dadurch geschützt sind, daß den einzelnen Speicherzellen je eine zusätzliche Kennung (Kontrollspeicher) zugeordnet ist.

Hierzu 1 Seite(n) Zeichnungen

– Leerseite –

Nummer: DE 42 21 278 A1
Int. Cl.[5]: G 06 F 15/16
Offenlegungstag: 5. Januar 1994

Fig. 1



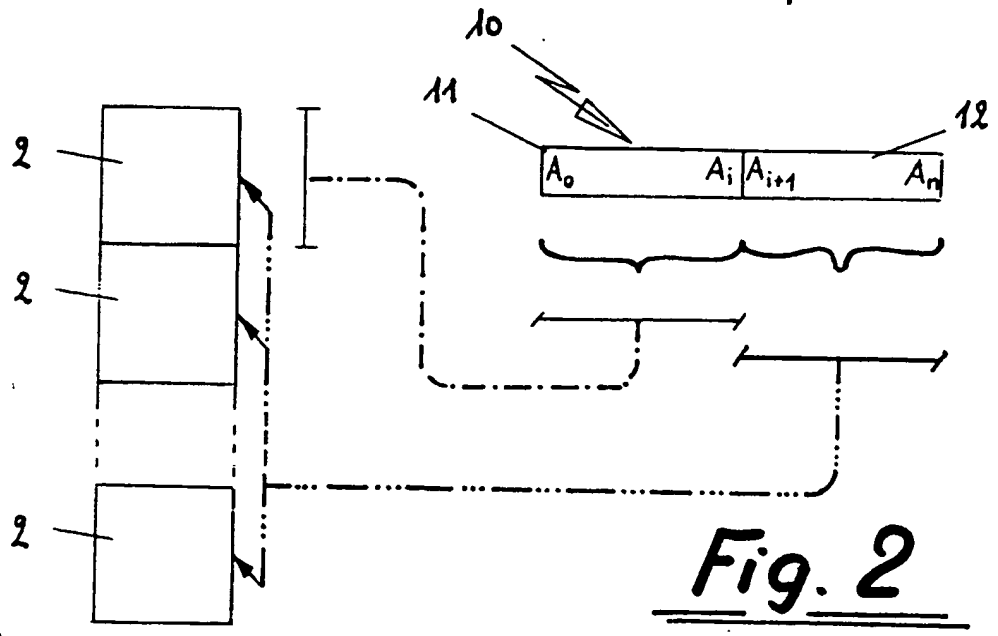
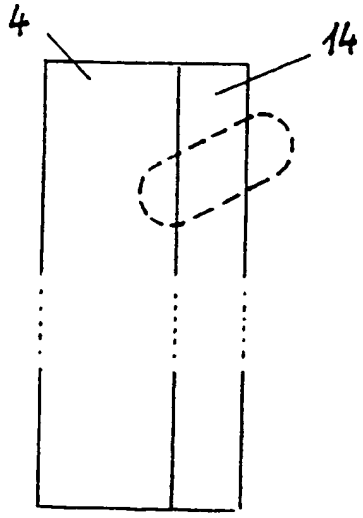
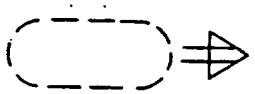


Fig. 2





0000000 a)

0111111 b)

0000010 c)

1010000 d)

Fig. 3